Abstract

CubeSats, small satellites with standardized dimensions and masses, have revolutionized space exploration with their cost-effectiveness and versatility. Accurate Attitude Determination and Control Systems (ADCS) are essential for CubeSats to maintain their desired orientation in space, enabling precise payload pointing, optimized power generation, reliable communication, and efficient orbital manoeuvres. This report presents the development and implementation of a comprehensive ADCS for CubeSat applications. The report begins with an introduction to CubeSats and the significance of ADCS in their operations. It outlines the project's objectives and motivation, emphasizing the importance of precise attitude control for mission success. A background and literature review section follows, discussing existing attitude determination and control methods, sensor technologies (e.g., magnetometers, sun sensors, IMUs), and actuator technologies (e.g., reaction wheels, magnetorquers). The core of the report focuses on the system architecture, implementation, and simulation/testing of the ADCS. The system architecture encompasses the integration of sensor and actuator technologies, along with sophisticated algorithms for attitude determination and control. The implementation details include software development in Python, describing the classes and functionalities for sensor data processing, attitude estimation, and control law computation. Simulations and testing results demonstrate the system's performance in accurately determining attitude and applying corrective control torques. In conclusion, the report summarizes the project achievements, highlights limitations and challenges, and suggests avenues for future work. Overall, this project contributes to advancing CubeSat technology by providing a robust ADCS solution that enhances small satellite capabilities and reliability for various space missions.

Chapter 1

1.1 Introduction

CubeSats or small satellites with standardized dimensions and masses, have emerged as a cost-effective and versatile platform for various space missions, ranging from technology demonstrations to scientific experiments and Earth observation tasks. As these miniaturized satellites become increasingly capable and widespread, the accurate determination and precise control of their attitude, or orientation, have become crucial aspects of their operations.

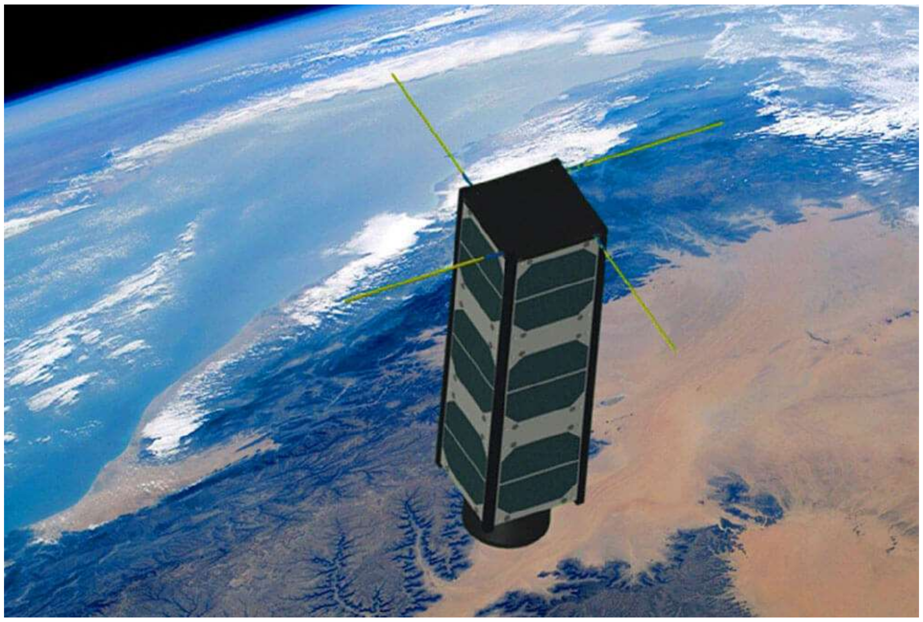


Figure 1 A CubeSat in Space

The Attitude Determination and Control System (ADCS) plays a vital role in ensuring that a CubeSat maintains the desired orientation throughout its mission lifetime. This system is responsible for estimating the satellite's current attitude based on sensor measurements and applying corrective torques to align the satellite with the required attitude. An efficient ADCS not only enables the CubeSat to achieve its intended objectives but also maximizes the utility of its onboard instruments and payloads.

1.2 Scope

This project aims to develop a comprehensive ADCS for CubeSat applications, encompassing both attitude determination and control subsystems. The attitude determination subsystem fuses data from multiple sensors, such as magnetometers, sun sensors, inertial measurement units (IMUs), and Global Positioning System (GPS) receivers, to estimate the satellite's orientation accurately. The attitude control subsystem employs actuators, such as reaction wheels and magnetorquers, to generate the necessary torques to maintain or adjust the CubeSat's attitude as required. The development of the ADCS involves the implementation of sensor fusion algorithms, attitude estimation techniques, control laws, and the integration of hardware components. Simulation and testing are crucial steps in evaluating the system's performance and ensuring its reliability before deployment in space missions. By addressing the challenges of attitude determination and control for CubeSats, this project contributes to the advancement of small satellite technology and its applications in various fields, including Earth observation, scientific research, and technology demonstration missions.

This report in organised into the following sections: Chapter 1 offers a brief yet informative overview of CubeSats, emphasizing their increasing significance in space missions due to their cost-effectiveness and versatility. Chapter 2 delves into the theoretical underpinnings of ADCS, elucidating why precise attitude determination and control are vital for CubeSat operations. The system architecture section in Chapter 3 provides a detailed breakdown of ADCS components and functionalities, outlining the integration of sensors, actuators, and algorithms. Chapter 4 explores the practical implementation of ADCS, focusing on software and hardware aspects. Results of simulations conducted to assess the performance of the implemented ADCS are presented in Chapter 5. In the conclusion, the key findings and contributions of the project are summarized. The section reflects on the strengths and limitations of the implemented ADCS, suggests areas for future improvement, and underscores the significance of the project in advancing CubeSat technology.

Chapter 2

Theoretical Perspective

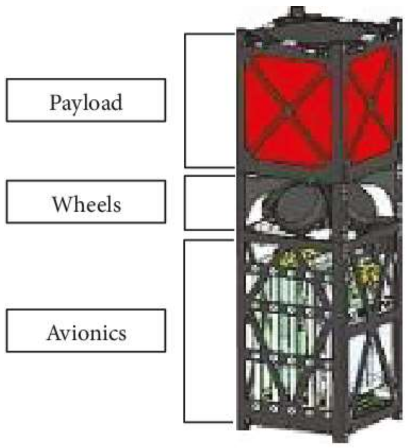


Figure 2.1 Structure of a CubeSat

2.1 Significance

Precise attitude determination and control are critical for the successful operation of CubeSats in space missions. The attitude of a satellite is defined as its orientation relative to a reference frame, typically an inertial or Earth-centred frame. Maintaining the desired attitude is essential for various reasons:

a. Payload Pointing
   Many CubeSat missions involve scientific instruments or payloads that require precise pointing towards targets of interest, such as celestial objects, specific regions on Earth, or communication links. Accurate attitude control ensures that these payloads are correctly oriented, maximizing their effectiveness and data quality.
b. Power Generation
   CubeSats typically rely on solar panels for power generation. By actively controlling the attitude, the satellite can optimize the orientation of its solar panels towards the Sun, maximizing the amount of energy harvested and extending the mission lifetime.
c. Communication
   Reliable communication between the CubeSat and ground stations is crucial for data transmission and command uplinks. Attitude

control ensures that the satellite's antennas are properly aligned with the desired ground station, maintaining a stable communication link.

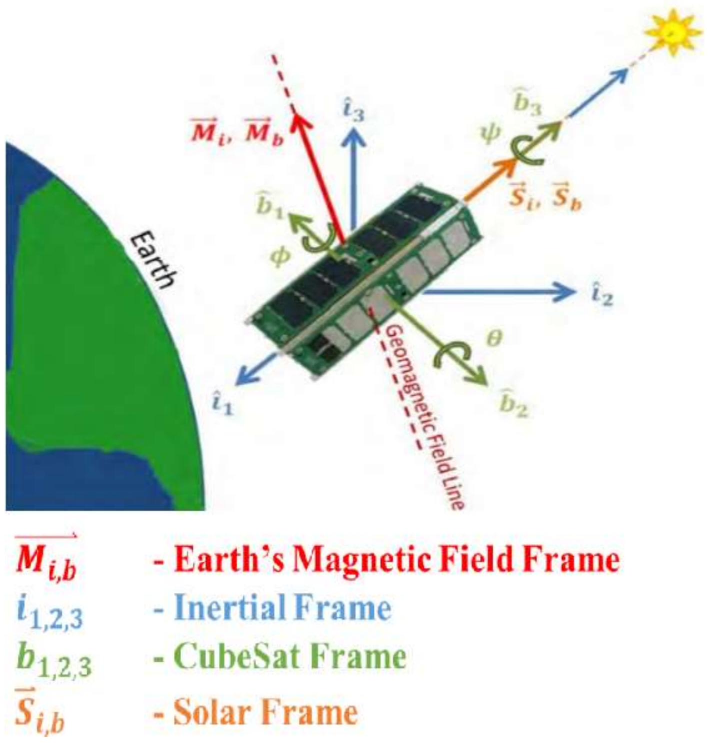


Figure 2.2 Different Frames of a CubeSat

2.2 ADCS System

Various methods and algorithms have been developed for attitude determination and control in CubeSats. Some common approaches include:

a. Kalman filtering techniques, such as the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF), for sensor fusion and state estimation.
b. QUEST (QUaternion ESTimation) algorithm for optimal attitude determination from vector observations.
c. Madgwick filter, a computationally efficient algorithm for orientation estimation using inertial and magnetic field data.
d. PID (Proportional-Integral-Derivative) control for attitude stabilization and tracking.
e. Quaternion feedback control for robust and efficient attitude maneuvering.

f. Sliding mode control for dealing with system uncertainties and disturbances.
g. Model Predictive Control (MPC) for optimal control considering constraints and system dynamics.

2.3 Sensors for Attitude Determination

Attitude determination relies on measurements from various onboard sensors. Common sensor technologies employed in CubeSats include:

a. Magnetometers
   Magnetometers measure the Earth's magnetic field vector relative to the satellite's body frame. By comparing the measured magnetic field with a reference model, the satellite's attitude can be partially determined.
b. Sun Sensors
   Sun sensors measure the direction of the Sun relative to the satellite's body frame. This information can be used to determine the attitude by comparing the measured Sun vector with its expected direction.
c. Inertial Measurement Units (IMUs)
   IMUs consist of accelerometers and gyroscopes that measure the satellite's linear accelerations and angular rates, respectively. IMU data can be integrated to estimate the attitude, but is typically combined with other sensor measurements due to drift accumulation.
d. Global Positioning System (GPS) Receivers
   While not primarily designed for attitude determination, GPS receivers can provide attitude information by analyzing the relative positions of multiple GPS satellites and their signals.

2.4 Attitude Corrections

To apply corrective torques and maintain the desired attitude, CubeSats typically employ the following actuator technologies:

a. Reaction Wheels
   Reaction wheels are momentum exchange devices that generate torques by accelerating or decelerating a spinning wheel. They

provide high-precision attitude control but can saturate over time due to accumulated angular momentum.

b. Magnetorquers

   Magnetorquers generate torques by interacting with the Earth's magnetic field. They are commonly used for momentum dumping and coarse attitude control but have limited torque authority.

c. Thrusters

   In some cases, small thrusters can be used for attitude control by generating thrust forces that create torques on the satellite. However, thrusters consume propellant and are typically reserved for larger attitude adjustments or orbital manoeuvres.

Chapter 3

System Architecture

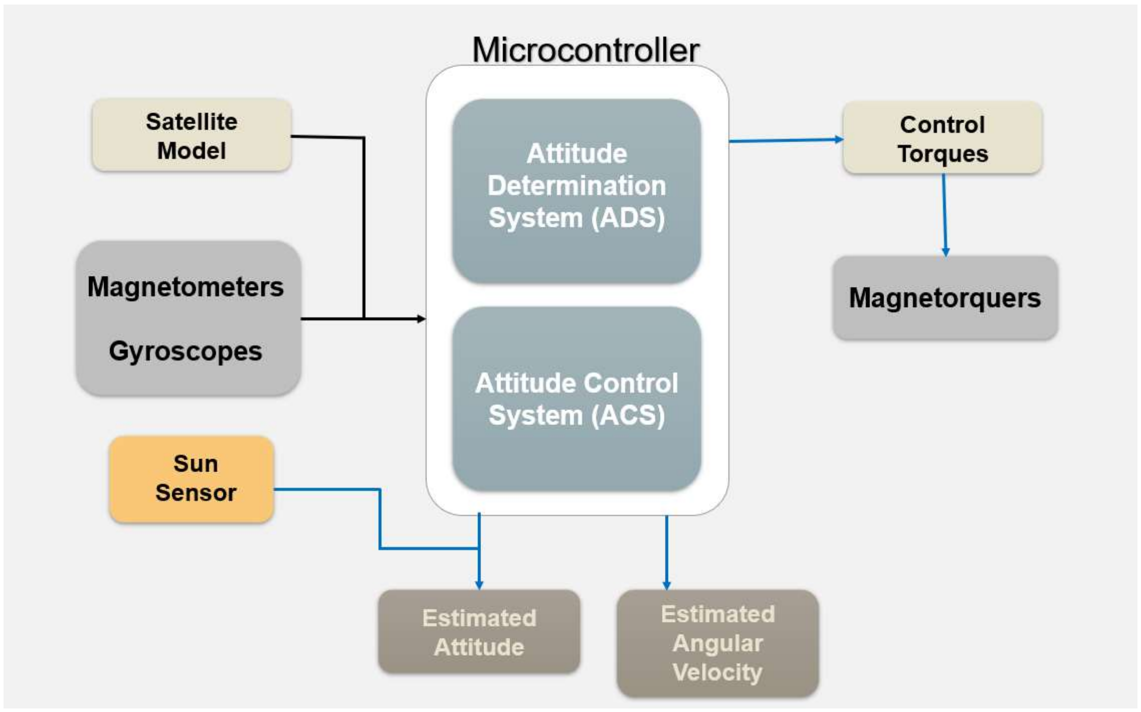


Figure 3 System Architecture of CubeSat ADCS

The system architecture of our CubeSat's Attitude Determination and Control System (ADCS) encompasses the integration of sensor technologies for attitude determination and actuator technologies for attitude control, along with the implementation of algorithms for sensor fusion and control law computation.

3.1 Attitude Determination System

The Attitude Determination System comprises various sensors responsible for measuring different aspects of the satellite's orientation. The sensors utilized include:

a. Magnetometer: Utilized for measuring the Earth's magnetic field, providing information about the satellite's orientation relative to the magnetic field vector.
b. Gyroscope (IMU): Used to measure angular rates, providing dynamic information about the satellite's rotational motion.

c. Sun Sensor: Provides data about the sun's position relative to the satellite, aiding in precise attitude determination in orbit.
d. GPS Receiver: Optionally employed for obtaining position information, which can supplement attitude determination algorithms.

These sensors collectively enable the estimation of the CubeSat's attitude through sensor fusion and estimation algorithms the Madgwick filter and complementary filter.

3.2 Attitude Control System

The Attitude Control System employs actuators to adjust the CubeSat's orientation based on the desired attitude computed by the control algorithms. The actuators utilized include:

a. Reaction Wheels: Act as momentum exchange devices, allowing for the adjustment of the satellite's orientation by varying their rotational speed.
b. Magnetorquers: Utilize the interaction between the Earth's magnetic field and onboard electromagnets to exert torque and adjust the satellite's attitude passively.

Control algorithms, PID control and quaternion feedback control, are implemented to compute the required torque commands for the actuators based on the deviation between the desired and current attitude states. These control algorithms are executed within attitude control loops, which continuously adjust the attitude based on sensor feedback.

Chapter 4

Implementation

The software implementation of the ADCS involves the development of Python classes to handle sensor data processing, attitude estimation, sensor fusion, and control law computation. The classes encapsulate the functionalities of each component, facilitating modularity and ease of integration. Additionally, hardware implementation details include the description of the CubeSat platform or testbed, sensor, and actuator integration, as well as communication interfaces and protocols.

The software implementation of the Attitude Determination and Control System (ADCS) for CubeSats is written in Python and consists of several modules and classes. The modular design allows for easy integration, testing, and maintenance of individual components.

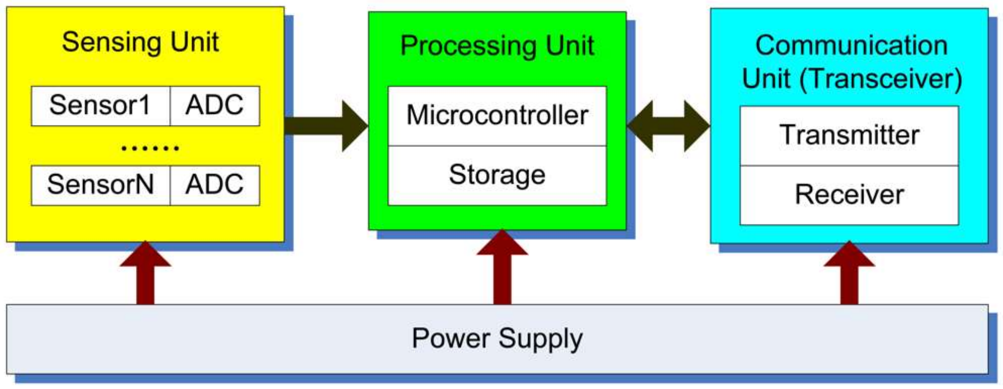


Figure 4 Various units of ADCS sub-system

4.1 Attitude Determination System (ADS)

The ADS is implemented in the `*AttitudeDeterminationSystem*` class, which handles sensor data acquisition, attitude estimation, and sensor fusion. The key components of the ADS implementation are:

a. Sensor Data Handling

The `*update_sensor_data*` method of the `*AttitudeDeterminationSystem*` class is responsible for updating the sensor data from various sources, such as magnetometers, IMUs, sun sensors, and GPS receivers. In the provided code, these sensor data are simulated or represented as dummy variables.

b. Attitude Estimation Algorithms
   The class implements several methods for estimating the satellite's attitude using different sensor data:
   i. `*estimate_magnetometer_attitude*`*:* Estimates the attitude quaternion based on magnetometer data, assuming the CubeSat's yaw is aligned with Earth's magnetic north.
   ii. `*estimate_imu_attitude*`: Estimates the attitude quaternion by integrating angular rates from the IMU data, using a simple complementary filter.
   iii. `*estimate_sun_sensor_attitude*`: Estimates the attitude quaternion from sun sensor data, assuming the sun is aligned with the positive z-axis in the CubeSat frame.

c. Sensor Fusion

The `*sensor_fusion*` method implements a simple averaging technique to fuse the attitude estimates from multiple sensors.

4.2 Attitude Control System (ACS)

a. Control Laws and Actuator Command Generation

The `*Actuation.py*` file contains an implementation of a PID (Proportional-Integral-Derivative) controller for attitude control. The controller generates actuator commands (torque commands) based on the error between the desired and current attitudes.

The main control loop iterates over time steps and performs the following tasks:

   i. Compute the quaternion error between the desired and current attitudes using the `quaternion_error` function.
   ii. Calculate the PID control terms (proportional, integral, and derivative) based on the quaternion error.
   iii. Generate actuator commands (torque commands) for reaction wheels and magnetorquers based on the control terms and actuator constraints.
   iv. Update the angular velocity and quaternion based on the applied torques (using simplified dynamics and kinematics).

b. Momentum Management

The code does not explicitly implement a momentum management strategy. However, the `*torque_cmd_mtq*` variable represents the torque command for magnetorquers, which is used for momentum dumping and desaturating reaction wheels.

c. Utility Functions

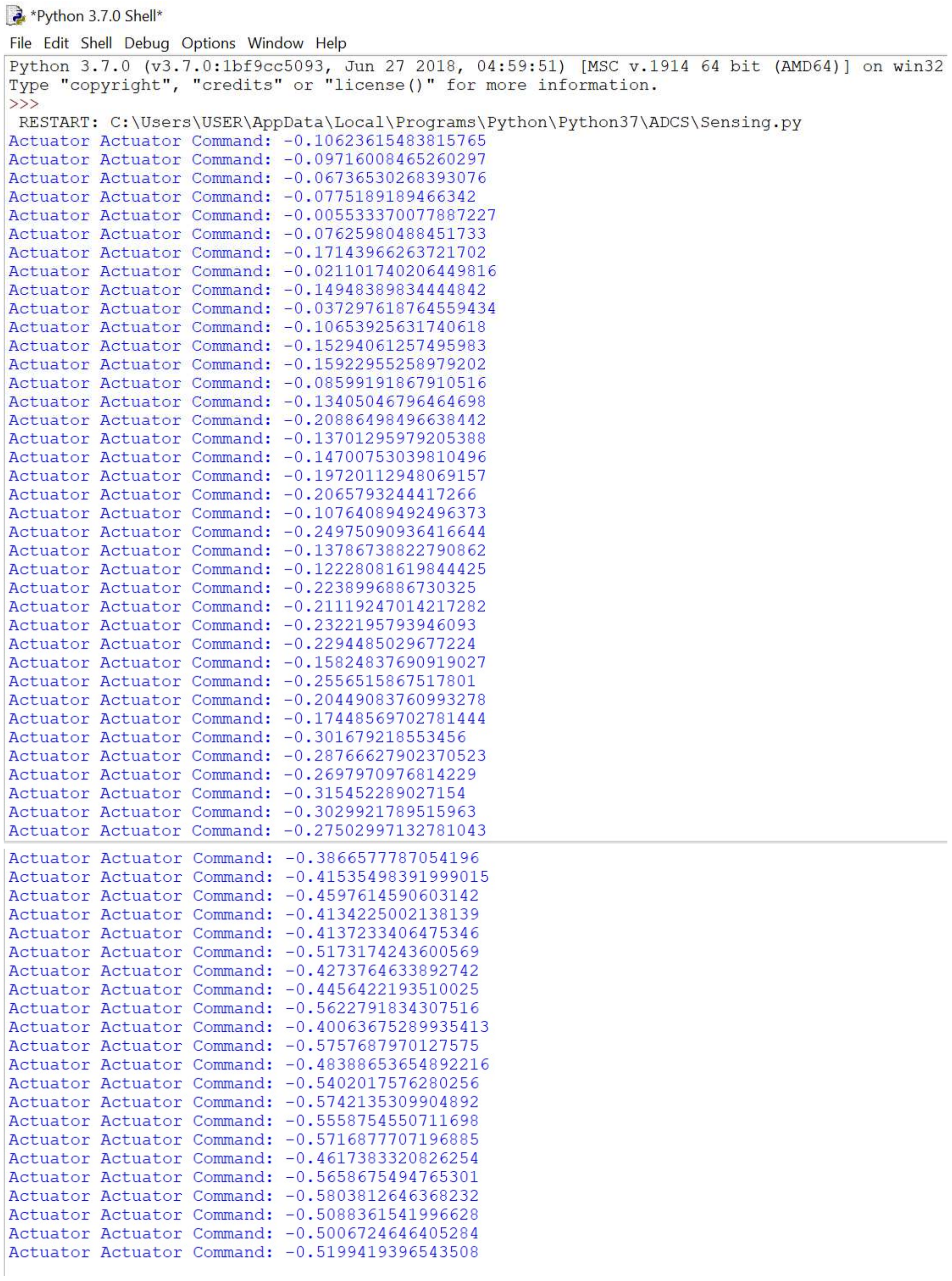
The code includes several utility functions for quaternion operations, such as quaternion multiplication, conversion between quaternions and Euler angles, and vector rotation using quaternions. These functions support the attitude estimation and control algorithms.

d. Simulation and Plotting

The code simulates the ADCS and plots the relevant data, such as quaternions, angular velocities, and actuator commands over time. These simulations are primarily for demonstration and testing purposes.

Chapter 5

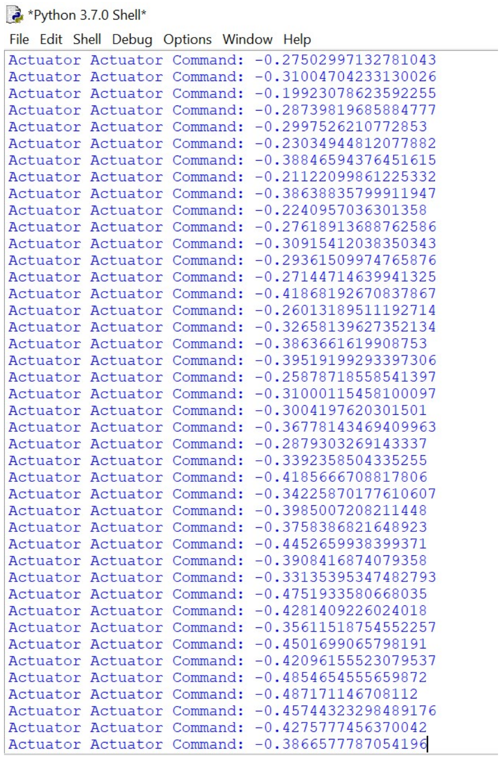
Simulations and Results



Python 3.7.0 Shell

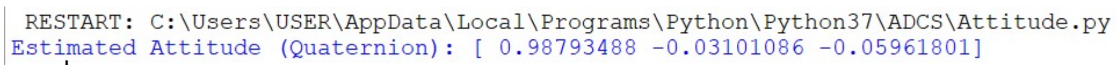
```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\USER\AppData\Local\Programs\Python\Python37\ADCS\Sensing.py
Actuator Actuator Command: -0.10623615483815765
Actuator Actuator Command: -0.09716008465260297
Actuator Actuator Command: -0.06736530268393076
Actuator Actuator Command: -0.0775189189466342
Actuator Actuator Command: -0.005533370077887227
Actuator Actuator Command: -0.07625980488451733
Actuator Actuator Command: -0.17143966263721702
Actuator Actuator Command: -0.021101740206449816
Actuator Actuator Command: -0.14948389834444842
Actuator Actuator Command: -0.037297618764559434
Actuator Actuator Command: -0.10653925631740618
Actuator Actuator Command: -0.15294061257495983
Actuator Actuator Command: -0.15922955258979202
Actuator Actuator Command: -0.08599191867910516
Actuator Actuator Command: -0.13405046796464698
Actuator Actuator Command: -0.20886498496638442
Actuator Actuator Command: -0.13701295979205388
Actuator Actuator Command: -0.14700753039810496
Actuator Actuator Command: -0.19720112948069157
Actuator Actuator Command: -0.2065793244417266
Actuator Actuator Command: -0.10764089492496373
Actuator Actuator Command: -0.24975090936416644
Actuator Actuator Command: -0.13786738822790862
Actuator Actuator Command: -0.12228081619844425
Actuator Actuator Command: -0.2238996886730325
Actuator Actuator Command: -0.21119247014217282
Actuator Actuator Command: -0.2322195793946093
Actuator Actuator Command: -0.2294485029677224
Actuator Actuator Command: -0.15824837690919027
Actuator Actuator Command: -0.2556515867517801
Actuator Actuator Command: -0.20449083760993278
Actuator Actuator Command: -0.17448569702781444
Actuator Actuator Command: -0.301679218553456
Actuator Actuator Command: -0.28766627902370523
Actuator Actuator Command: -0.2697970976814229
Actuator Actuator Command: -0.315452289027154
Actuator Actuator Command: -0.3029921789515963
Actuator Actuator Command: -0.27502997132781043
```

```
Actuator Actuator Command: -0.3866577787054196
Actuator Actuator Command: -0.41535498391999015
Actuator Actuator Command: -0.4597614590603142
Actuator Actuator Command: -0.4134225002138139
Actuator Actuator Command: -0.4137233406475346
Actuator Actuator Command: -0.5173174243600569
Actuator Actuator Command: -0.4273764633892742
Actuator Actuator Command: -0.4456422193510025
Actuator Actuator Command: -0.5622791834307516
Actuator Actuator Command: -0.40063675289935413
Actuator Actuator Command: -0.5757687970127575
Actuator Actuator Command: -0.48388653654892216
Actuator Actuator Command: -0.5402017576280256
Actuator Actuator Command: -0.5742135309904892
Actuator Actuator Command: -0.5558754550711698
Actuator Actuator Command: -0.5716877707196885
Actuator Actuator Command: -0.4617383320826254
Actuator Actuator Command: -0.5658675494765301
Actuator Actuator Command: -0.5803812646368232
Actuator Actuator Command: -0.5088361541996628
Actuator Actuator Command: -0.5006724646405284
Actuator Actuator Command: -0.5199419396543508
```



```
Actuator Actuator Command: -0.27502997132781043
Actuator Actuator Command: -0.31004704233130026
Actuator Actuator Command: -0.19923078623592255
Actuator Actuator Command: -0.28739819685884777
Actuator Actuator Command: -0.2997526210772853
Actuator Actuator Command: -0.23034944812077882
Actuator Actuator Command: -0.38846594376451615
Actuator Actuator Command: -0.21122099861225332
Actuator Actuator Command: -0.38638835799911947
Actuator Actuator Command: -0.2240957036301358
Actuator Actuator Command: -0.27618913688762586
Actuator Actuator Command: -0.30915412038350343
Actuator Actuator Command: -0.29361509974765876
Actuator Actuator Command: -0.27144714639941325
Actuator Actuator Command: -0.41868192670837867
Actuator Actuator Command: -0.26013189511192714
Actuator Actuator Command: -0.32658139627352134
Actuator Actuator Command: -0.3863661619908753
Actuator Actuator Command: -0.39519199293397306
Actuator Actuator Command: -0.25878718558541397
Actuator Actuator Command: -0.31000115458100097
Actuator Actuator Command: -0.3004197620301501
Actuator Actuator Command: -0.36778143469409963
Actuator Actuator Command: -0.2879303269143337
Actuator Actuator Command: -0.3392358504335255
Actuator Actuator Command: -0.4185666708817806
Actuator Actuator Command: -0.34225870177610607
Actuator Actuator Command: -0.3985007208211448
Actuator Actuator Command: -0.3758386821648923
Actuator Actuator Command: -0.4452659938399371
Actuator Actuator Command: -0.3908416874079358
Actuator Actuator Command: -0.33135395347482793
Actuator Actuator Command: -0.4751933580668035
Actuator Actuator Command: -0.4281409226024018
Actuator Actuator Command: -0.35611518754552257
Actuator Actuator Command: -0.4501699065798191
Actuator Actuator Command: -0.42096155523079537
Actuator Actuator Command: -0.4854654555659872
Actuator Actuator Command: -0.487171146708112
Actuator Actuator Command: -0.45744323298489176
Actuator Actuator Command: -0.4275777456370042
Actuator Actuator Command: -0.3866577787054196
```

Figure 5.1 Output of Actuators



```
 RESTART: C:\Users\USER\AppData\Local\Programs\Python\Python37\ADCS\Attitude.py
Estimated Attitude (Quaternion): [ 0.98793488 -0.03101086 -0.05961801]
```

Figure 5.2 Output of Attitude Determination Unit

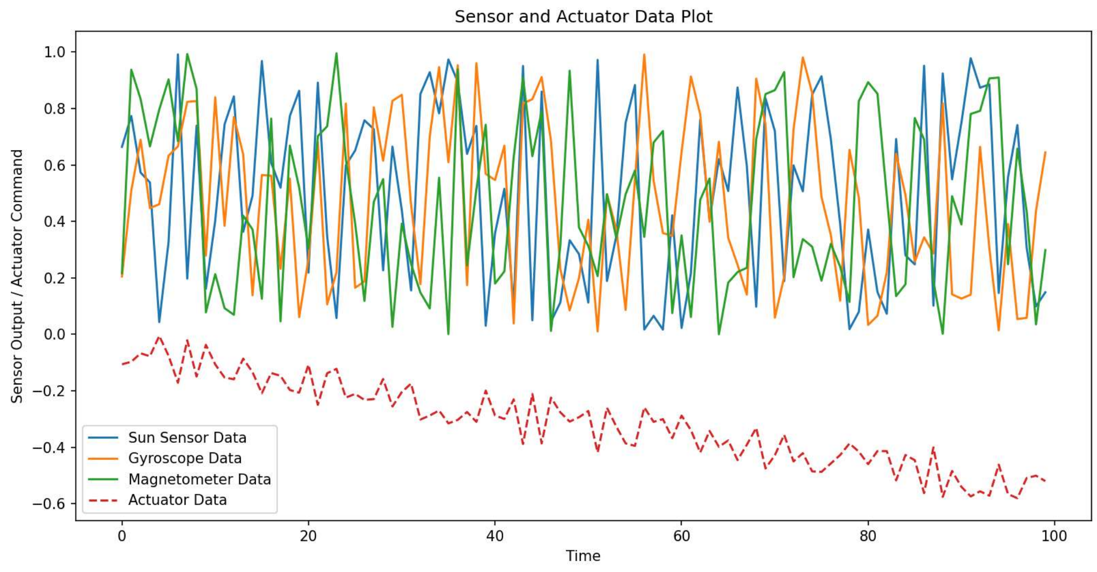
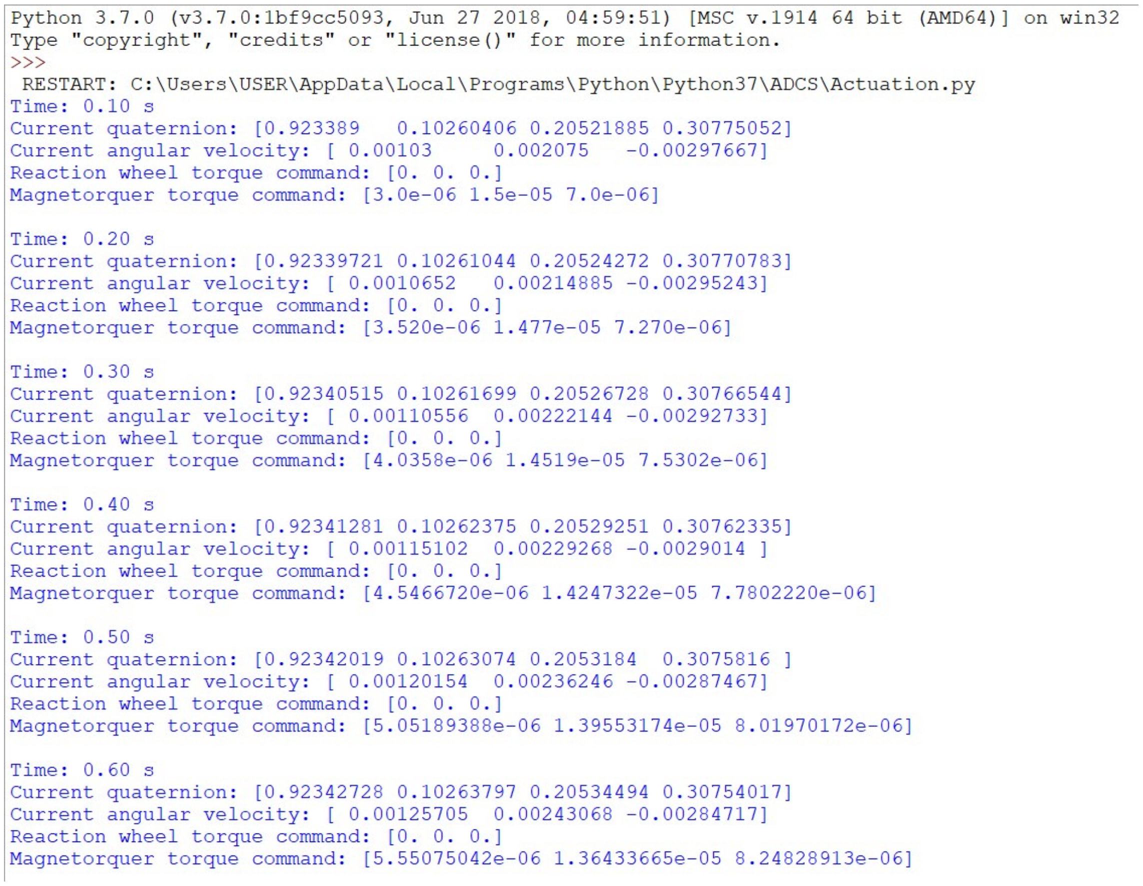


Figure 5.3 Plot of Sensor and Actuator Data



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\USER\AppData\Local\Programs\Python\Python37\ADCS\Actuation.py
Time: 0.10 s
Current quaternion: [0.923389   0.10260406 0.20521885 0.30775052]
Current angular velocity: [ 0.00103     0.002075   -0.00297667]
Reaction wheel torque command: [0. 0. 0.]
Magnetorquer torque command: [3.0e-06 1.5e-05 7.0e-06]

Time: 0.20 s
Current quaternion: [0.92339721 0.10261044 0.20524272 0.30770783]
Current angular velocity: [ 0.0010652   0.00214885 -0.00295243]
Reaction wheel torque command: [0. 0. 0.]
Magnetorquer torque command: [3.520e-06 1.477e-05 7.270e-06]

Time: 0.30 s
Current quaternion: [0.92340515 0.10261699 0.20526728 0.30766544]
Current angular velocity: [ 0.00110556  0.00222144 -0.00292733]
Reaction wheel torque command: [0. 0. 0.]
Magnetorquer torque command: [4.0358e-06 1.4519e-05 7.5302e-06]

Time: 0.40 s
Current quaternion: [0.92341281 0.10262375 0.20529251 0.30762335]
Current angular velocity: [ 0.00115102  0.00229268 -0.0029014 ]
Reaction wheel torque command: [0. 0. 0.]
Magnetorquer torque command: [4.5466720e-06 1.4247322e-05 7.7802220e-06]

Time: 0.50 s
Current quaternion: [0.92342019 0.10263074 0.2053184  0.3075816 ]
Current angular velocity: [ 0.00120154  0.00236246 -0.00287467]
Reaction wheel torque command: [0. 0. 0.]
Magnetorquer torque command: [5.05189388e-06 1.39553174e-05 8.01970172e-06]

Time: 0.60 s
Current quaternion: [0.92342728 0.10263797 0.20534494 0.30754017]
Current angular velocity: [ 0.00125705  0.00243068 -0.00284717]
Reaction wheel torque command: [0. 0. 0.]
Magnetorquer torque command: [5.55075042e-06 1.36433665e-05 8.24828913e-06]
```

Figure 5.4 Output of Magnetorquer

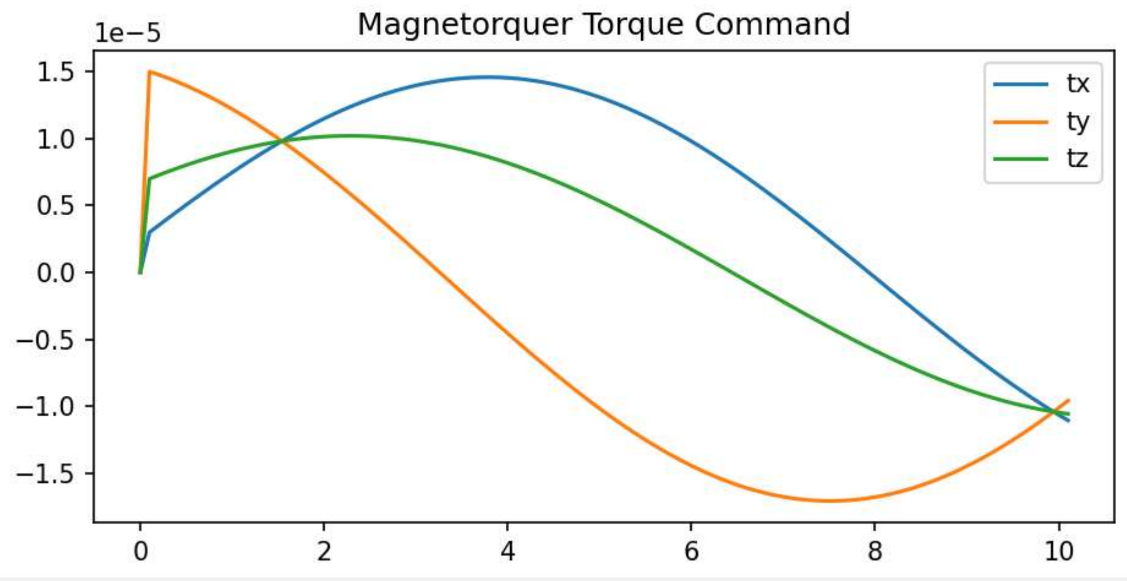


Figure 5.5 Plot of Magnetorquer Command

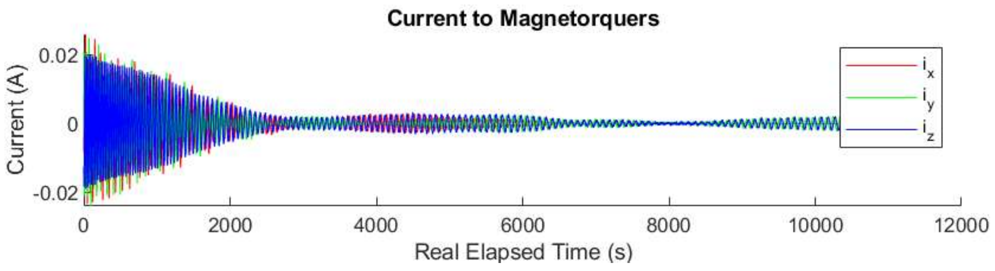


Figure 5.6 Plot of Magnetorquers Current Input

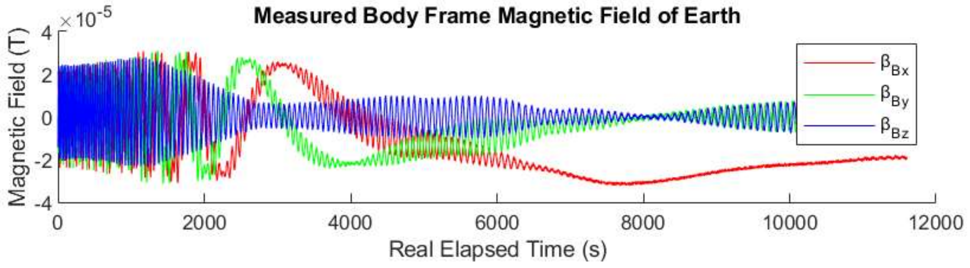


Figure 5.7 Plot of Earth`s Magnetic Field

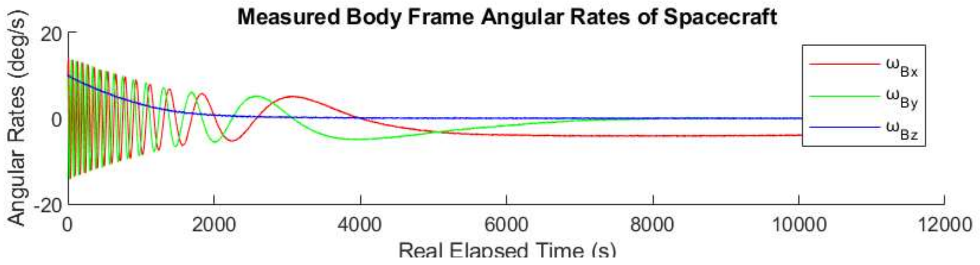


Figure 5.8 Plot of CubeSat`s Angular Rates